**CDAC Feb 2015**

16 x 4 Priority Encoder

Name: Bhrigu Bhargava

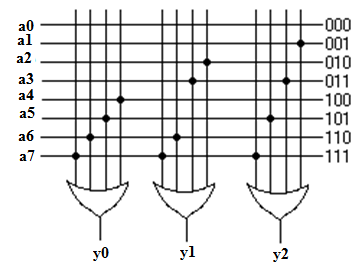
**16 x 4 Priority Encoder using Concurrent Statements**

# Design Approach:

Digital Encoder is a Digital Device opposite of the Digital Decoder. A simple encoder circuit can receive a single active input out of 2n input lines generate a binary code on n parallel output lines. For example a single bit 16 to 4 encoder takes in 16 bits input) and 4 outputs .

A regular encoder assumes that exactly one input is 1 (while all other inputs are 0). Consider this assumption. Normally, we have 16 inputs, which mean we have 28 or 256 rows. Only 16 rows satisfy the assumption. The truth table for a 16-4 binary encoder (16 inputs and 4 outputs) is shown below. It is assumed that only one input has a value of 1 at any given time, and then the output is invalid.

# Circuit Diagram:

****

**Fig 1:- Circuit Diagram of a Priority Encoder**

**Truth Table:**

Example using 8x3 encoder:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Input | | | | | | | | Output | | |
| Y0 | **Y1** | **Y2** | **Y3** | **Y4** | **Y5** | **Y6** | **Y7** | **a** | **b** | **c** |
| 1 | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** |
| 0 | **1** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **1** |
| 0 | **0** | **1** | **0** | **0** | **0** | **0** | **0** | **0** | **1** | **0** |
| 0 | **0** | **0** | **1** | **0** | **0** | **0** | **0** | **0** | **1** | **1** |
| 0 | **0** | **0** | **0** | **1** | **0** | **0** | **0** | **1** | **0** | **0** |
| 0 | **0** | **0** | **0** | **0** | **1** | **0** | **0** | **1** | **0** | **1** |
| 0 | **0** | **0** | **0** | **0** | **0** | **1** | **0** | **1** | **1** | **0** |
| 0 | **0** | **0** | **0** | **0** | **0** | **0** | **1** | **1** | **1** | **1** |

**Source Code:**

module pe (output reg [3:0] op, output reg valid, input [15:0] ip);

reg [4:0] temp1;

integer temp2=16;

//always @ (\*)

//begin

// while (temp1===5'bxxxxx)

// begin

//if (ip[temp2-1]==1)

// temp1={temp2-1,1'b1};

//else

// temp2=temp2-1;

//end

always @ ip

begin

if (ip[15]==1'b1)

temp1=5'b11111;

else if (ip[14]==1'b1)

temp1=5'b11101;

else if (ip[13]==1'b1)

temp1=5'b11011;

else if (ip[12]==1'b1)

temp1=5'b11001;

else if (ip[11]==1'b1)

temp1=5'b10111;

else if (ip[10]==1'b1)

temp1=5'b10101;

else if (ip[9]==1'b1)

temp1=5'b10011;

else if (ip[8]==1'b1)

temp1=5'b10001;

else if (ip[7]==1'b1)

temp1=5'b01111;

else if (ip[6]==1'b1)

temp1=5'b01101;

else if (ip[5]==1'b1)

temp1=5'b01011;

else if (ip[4]==1'b1)

temp1=5'b01001;

else if (ip[3]==1'b1)

temp1=5'b00111;

else if (ip[2]==1'b1)

temp1=5'b00101;

else if (ip[1]==1'b1)

temp1=5'b00011;

else if (ip[0]==1'b1)

temp1=5'b00001;

op=temp1[4:1];

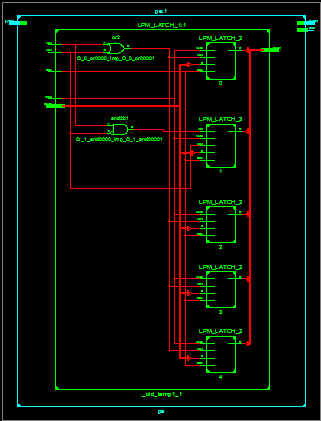
valid=temp1[0];

end

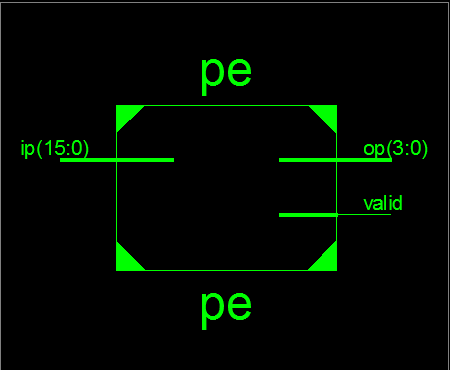
endmodule

**Synthesis:**

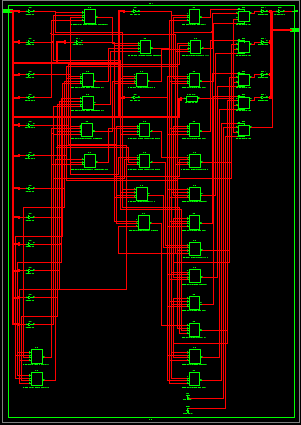
1. RTL Schematic



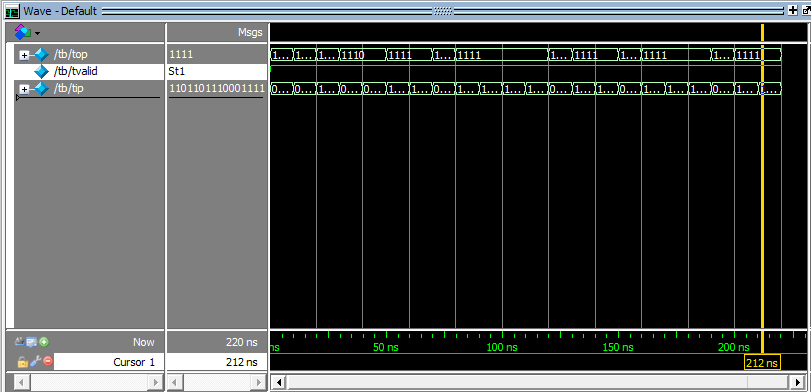
1. Block Diagram

****

1. Tech Schematic



1. Simulation Waveform Result

****

**Error:**

Tried alternative method but the previous value wasn’t overwritten.

Soln: Went for bit by bit analysis.

**Verified by:**

Dharamvir Chundavat (150240133007)